Then when they put in the networking stuff, that's when I changed. There was a guy at Berkeley who was named Eric Allman, who wrote the Mailer, which runs most of the Internet mail. He would probably remember these things because he had written a version of that program before the standard emerged. His Mailer still handles a lot of mail to corporate networks and other networks that don't have the Internet standard addresses. So just the adoption of an Internet standard way of addressing the SON.COM and all of this, that, and the other. That was quite enormous. It's amazing how somebody sold PET.COM for $250,000 that they had registered for nothing. If I had realized last year, I could have just probably registered 1,000 names and then sold them. The going rate is $25,000 to $50,000 a piece. If you had DOG.COM or CAT.COM, it's like 1-800-Florist or 1-800-Flowers. These things were available for free. Now we registered a bunch for other people, but I didn't pick up on that. We should have just registered like DOGFOOD.COM. It's completely goofy.

[Agrees.]

You should stop the interview right now and rush out and register. Was it 1977? What year did they switch to the TCP/IP and the Internet? '79?

Yeah. So let's go back.

That's when they turned the old one off.

That's when they turned NCP off and turned TCP/IP on.

Was it that late. I thought it was in the 70's. That shows you I don't know dates. I have the *"I survived the TCP/IP"* transition button at home somewhere. It has the date on it. That Dan Lynch gave me at one point.

I think I started to mention this to you on the phone that Berkeley was never really on the Net. I was, in fact, doing an introduction at Asuna Sewing Machines and I was standing on the stage with this little cart that they had wheeled out four machines, and there was more computing power on this cart than there was in the

entire Internet in '82. There were only a few hundred machines. There were a few MIPs each or whatever.

But what happened was Berkeley had never really been on the Net, and when I came to Berkeley, people would get on the Arpanet by either logging into Illiac down at NASA Ames or eventually the Ingress. I think the Ingress machine got on the Internet. But what happened was most of the machines on the Internet were PDP-10's. Places like Stanford, CMU, MIT, and ISI. There were like four machines at MIT and those were really the cornerstone machines on the Internet. They were KA's, KL's, KI's, you know, the big PDP-10's, and they were all running NCP. Berkeley got some money somehow to buy a new computer and there had to be a decision as to what kind of computer to get. It was very late in the day to be in the PDP-10 set. They all had their own operating systems, their own versions of LISP and doing their AI research. So the choice came down to be getting some small PDP-10 or to get this new machine called the Vax. What happened was the department decided to get a Vax which was the genesis then of taking Unix off the PDP-11 and putting it on the Vax.

Do you remember when this was?

No.

Were you a graduate student?

Yeah. I was a graduate student from '75 to '82. So that was probably '78 or '79. Then we took that and we modified Unix and we made it have virtual memory, and about maybe a year or two later, the people at Deck gave us a discount and gave us a whole bunch of Vax 750's which we could use like personal work stations. That was probably in '79, '80. At the same time, we got from BBN a machine. It was one of the C machines, which was an IMP that could be on either NCP or TCP Arpanet.

Well, they brought that in, and because of a contract we had with Darpa to do Unix for their researchers, they gave us one of these machines. What happened was we had this wide area network and some funny hosted after, but we didn't really have a network between the different computers and they really weren't good local area network interfaces for Unix machines.

So what we did was we had Dave Boggs at Xerox, who had a summer student who had implemented a Unibus Ethernet card at three megabits, and Dave agreed to make us some more. So I went down there in the middle of the night. We had procured the parts. We gave him some money. He made the boards. We took them to the lab. It was 3:00 in the morning. He debugged the cards on this oscilloscope to make sure they were working. I'd never basically been to Stanford before. I got in my car and got lost in Atherton somewhere trying to find a freeway in the middle of the night trying to get home. But we brought these cards back to Berkeley and stuck them in our machines and that gave us a three megabit Ethernet between the different machines.

Then what I did was I got the source code for the first version of TCP/IP that had been done at BBN and tried to make it work better on the three megabit cards. That was the beginnings.

By this point you had already done Unix BSD?

Right. There already were BSD's. There had been 1 and 2 BSD, which were tapes for the PDP-11. 3 was the first system that was a complete operating system. So you got the tape and you could actually start with just the hardware and it's like a boot disk. 3 had virtual memory. Then Version 4, and there were several, 4-1, 4-2, whatever. That's when networking was added. Some of the commands that still exist are remote login. Some of these commands I added to what was called 4, and 4 also was the one that had the networking design in it, and that networking design was debugged using the Arpanet with the BBN machine, and also then on these three megabit Ethernet cards that we got. So that's what we used as a hardware environment for testing and getting the various software to work.

Until Mosaic and the WEB came, that was about all there was. There was a period of about at least 10 years from like '80 or '81 till '93, where Rlogin, Remote Shell, Telnet, FTP, and then the file sharing protocol. I did one when we were at Sun. Those were really the protocols that dominated the traffic on the Internet.

Tell me if you remember the first time you learned about TCP/IP.

I know the history now. I know it was sort of at Stanford and contemporaneously

with the Xerox protocols. We were aware of networking and knew we wanted to put it in on the system at Berkeley. We were also aware of some work that I think was really better than the protocols that we ended up with. It was done at Livermore by a guy named Dick Watson, called Delta T.

Why did you think you wanted networking in the protocols?

We wanted to do high speed wide area networking between research sites. You know, we had the network as the library. All these ideas were clearly ideas that we all wanted to be able to collaborate at long distance to take just to different buildings. Berkeley's in two buildings that are physically distance. We had been running serial lines between the buildings and running networks over serial lines. Eric had written a serial line network. It was really painful. You know, it's like batch cued. You want to copy your job to the computer center to go on the typesetter and it sits in the cue for four hours because the thing runs at 2400 bod. That was all really painful. So we wanted the networking.

Couldn't you just use the Arpanet?

The Arpanet wasn't that fast either. In some sense, we benefitted by being a backwash.

You were saying you didn't have an Arpanet node -- .

We didn't have that and we didn't have a PDP-10. Therefore, we could do the Vax and we could think about --. Now at the time, people were saying that you really needed broad band networks to run efficiently. We didn't believe that. So when we got this code and we got these three megabit interfaces up, the first thing we wanted to do was to make it run the three megabits. So that involved rewriting the code, which BBN had done, which quite frankly wasn't very well written in terms of being high performance, because they had only ever tested it on the 56K lines that they had that were the Internet or the NCP kind of environment.

They had done what?

Their code was fast enough and good enough to work in the WAN environment, but

had never been tested in a LAN environment or anything that was fast. Because the Ethernet at three megabits was 60 times faster than the long distance network, and then we found all sorts of bugs in the implementation and inefficiencies. I also had wanted to run more protocols than just TCP/IP. So what we did was I rewrote the code and redesigned the whole Unix system to not only handle things like Decnet and XNS and some of the other protocols that were out at the time, but also to take the code for TCP/IP and structure it in a way that was efficient so that it could run the three megabit Ethernet at full speed. I guess it was actually a 2.97 megabit Ethernet. I mean, that seems like a small difference of 1%. We were looking for every percent so we had the oscilloscopes out and stuff looking at what's going on.

You had a contract with Darpa?

We started doing it without the contract.

Then you got the Darpa money?

Well, Darpa had been burned by the PDP-10 community in that both the super computer community and the PDP-10 community -- every site kind of had their own operating system. In the ag community, every site had their own version of LISP. It was kind of frustrating for them because they couldn't -- the research results weren't portable between the sites. So they had been looking for a system they could use that would be in common between different sites and one that might even survive multiple generations of hardware. They created the network so that the people could communicate and because of their vision of using it for military use. But they needed a relatively more portable software solution. So for them, the notion of having -- they had like six operating systems for one instruction set and the PDP-10 from Deck, because there was the Stanford operating system and then there were two versions of Tenex and there was the ITS operating system from MIT and who knows how many versions of each of those. So it was really a terrible situation where even though they had had the same machine from the same vendor deck, you couldn't take a program off a machine in California and run it on a machine in Massachusetts. The program monitors were wanting to imagine having more than four universities doing stuff and they wanted to imagine that they could run the apps. So instead of four operating systems for one computer, we were looking for one operating system for four computers. Like an effective 16 benefit.

So Unix had already come from the PDP-11 to the Vax and they imagined that it could run in microprocessors and other things. You know, it's now 20-25 years old. So you could write software that could run on machines which hadn't been built yet and it would still work. That was the idea. So that was a big deal at the time. TCP/IP was designed to be scalable glue that would go beyond just running the relatively small network that they had done. They had lots of experiments back then with mobile packet radio. You think of digital cellular. And there was some experimental satellite stuff that was transcontinental. There was also one very early that went to Europe, to Norway or something. I remember people testing. Those gave you very acidic environments in which to try to do experiments because the packet round trip times would vary. The packet loss rates would vary and you could actually do experiments on how the network would behave in these kind of extreme conditions. It's kind of like sending something to the north or south pole for extreme weather testing. You know, expose it to wind chill. I guess there's probably a limit to how much wind chill you can get. But you don't find these things out until you actually try it. They built those experimental test beds basically by 1980 for sure.

That was sort of the origins of the thinking about Internetting was the packet radio.

Well, Xerox had always been an Internet.

[Skipped portion you requested.]

This was Novell's implementation of XNS?

XNS, right. So there were no routers. I mean, we invented our own routing protocol, and see, we didn't ask permission because they had the model that only BBN could do routing.

They?

Arpa.

Had the model that only BBN could do the routing?

Could do routers. So if I wanted to build a gateway like what Cisco eventually

became, I had to use BBN machines, but they were too slow because they were running this poorly implemented TCP.

These were the IMPs and the TIPs?

Right. You couldn't put them in an Ethernet environment. They were way too slow. They could switch 56K. So we just said, okay, well packet switching. That's exciting. That's like a packet comes into my Vax and it has two Ethernets and I have a table that has three entries and I look and decide what to do. That's like one page of code. We can do that. So we just made our computers into switches. Basically, Sun missed this whole systems switching market, because we never considered switching to be very big of a deal. We could just do it in the work stations. I didn't consider it to be worthy of a separate box.

They were doing some things that hadn't been done in terms of routing and putting routing tables in.

Right. That was all fine. The only point for us was that they were doing it in the context of 56K switch networks, and since we were soon 60 and then 200 times faster with three and ten megabit Ethernet, we had to not use their -- . So you could talk about the Ethernet being open and all, but when I got involved with it, the truth was that people were very concerned that BBN controlled the routing and that all the machines that switched the packets be run by BBN, which was really not tenable.

This was the mind set at the time that BBN had a control oriented mentality at the time.

Very parochial.

They had their network operations center in Cambridge.

I put up basically something sort of like a file server demon on their machine in Massachusetts so that I could compile programs and keep the source on both ends. They considered this to be a disaster because clearly sending files across the network like that would overload the network. People didn't have the vision of maybe we should be finding out what we can do if we use more bandwidth like multimedia traffic, file traffic, video voice. That was all really not kosher at all in the time

frame. This would have been like '78-'79.

So what were you doing?

So they got really mad at me. I just logged into their machine using Telnet. Started a piece of software which let me access files transparently across the country through the network. Just like a 10 page program. It's a great idea, right? But see, there's only 56K bandwidth and clearly my program accessing a file could just take all of it. Their motto was people should only type and use it as terminal emulation, because otherwise it would get too clogged.

What about FTP?

I don't know. I couldn't see what the difference was between --. The thing is FTP was inconvenient. By making it convenient, I would get people to use more.

And that was against --. I see.

Same thing about Mosaic. If you were a network administrator you could say this is a disaster. Now that we've made it easy for people to get information, the network is going to be flooded. We've got to turn this off. That was the kind of attitude they had.

So you were a graduate student in computer science or electrical engineering at Berkeley.

ECS. I had my little business which was I took a bunch of software we had written and had everyone send us $50 for the first version. I got 500 people that sent me $50 for the 1BSD and 2BSD. So we had $25,000 or $50,000 in a slush fund. We got to Frank. We had the departmental franking so we could send -- we had no postage expenses. We had to buy mag tapes and do some copy expense, but we had free use of the copy machine. So basically we had a nice little business. All the money went into the university account. We spent it on flying around and telephone calls. But we were having a lot of fun. Then we started sending out Unix for the Vax and charged people several hundred dollars, because you sent them like one case of eggs, like you see. It's like a cubic foot of stuff. Then eventually Bob Favory(?) got Darpa to basically give us money to do it in a more formal way. So we had specific

objectives.

But we pretty much did what we were doing all along, which was we were trying to make the system better. Because when we got the Vax and we got Unix for the Vax, it really wasn't at the same level of quality that VMS was in the Vax or certainly wasn't capable of supporting the research applications that people had. The file system wasn't fast enough. It didn't have virtual memory and it especially didn't have networking. So we wanted to put those things on and also graphically user interface good nap displays. We wanted to put those things on the system.

But we had already had this pattern of sending the stuff out. So we took the BBN code for TCP/IP and I just rewrote the whole thing. We basically sent it out on the Berkeley tape. That was free. That was just source code that was available to everybody and really it became the reference code for TCP/IP that everyone else used because it was very high quality and it wasn't really commercially constrained in any way. So anybody who wanted to build any sort of device with TCP/IP in it could grab this C code and then have a fully debugged. This thing had been tested both on LANs and WANs and it was very fast. Also it had the framework for supporting other protocols.

So how hard was it to put TCP/IP on Unix?

I already had an architecture for the structure of communications with idea -- what now called sockets. I think it's in Windows 95 even now. I invented sockets back '79 or '80. It was just sort of a mathematical model of how the network worked. Send a packet or make a connection and have the streams in some of the events. I was kind of abstract. So once you had that framework and you wrote all that framework for Unix, not making the communications look like a file, but making it look like an abstraction of communications, then taking it and making kind of a driver for XNS or for DECNET or whatever or ISO protocols or TCP was reasonably easy. Maybe 25 pages of C code. It wasn't easy to write the first one. But then if you know the protocol comes along, it has this particular -- . They are all pretty much the same. It's like romance languages or something, so you just make a copy and do it. So the first one took maybe six months of work by the time we were done and running a few drivers. But after that, a lot of people did it without a lot of struggle.

Was there any question whether you would throw your support behind TCP/IP versus OSI?

The choices were Delta T and XNS. Xerox was being really weird at the time. Like we asked them if we could buy a printer and they told us they wouldn't tell us how to print to it unless we bought a Star work station. So that was not going to work. Delta T, which I think are technically better protocols, required some synchronization of clocks, which required a policy that you would have to agree at all the sites that they would synchronize their time. There was no way I could get it. Didn't have a logistical way. So it made it harder to deploy. Although we would have been much better off if we had deployed it in the end. It didn't seem logistically reasonable to do it. We were always almost implementing it. We came so close to implementing it so many times and never quite did. Although they did use it at the Super Computer Center there in Livermore.

You mean Delta T?

Uh-huh. But that was the typical thing like I talked about before. Every PDP-10 site or Super Computer site had their own OS, their own language, and in this case, also their own network. So this is another example of it's kind of the last of that. But TCP is good enough. The technical problem that the other one solved hasn't really hurt us that much over the last 15 years.

So then when you went to Sun, was it in your head that all the networking stuff should go too?

The Unix work stations looked just like the Vax 750s that they had given us. Ethernet. They were smaller because they had a microprocessor. They had virtual memory. So it was basically just a shrink of the Vaxes. We were at the time running Unix on the Vax with Ethernet. So it was pretty straightforward to port to our work stations. Unix was portable. We just needed a C compiler for the different hardware.

And there was TCP in these?

There was TCP in the system at the point I left Berkeley, yeah. Although for the

first few months of Sun, we shipped a version of Unix that wasn't the Berkeley version, It didn't have the TCP in it. We soon switched to the Berkeley version.

So I guess I'm asking, do you think I'd be on the mark if I wrote in the book that it was Sun and the proliferation of the Sun work stations with TCP in them that gave way to the proliferation of standardization and implementation?

Oh, yeah. Basically if you look at, I don't know what year Dec, Intel and HP announced Ethernet and the open licensing of that. That probably was after Sun started. I don't remember. So the notion that networks would be open in that way was really kind of a controversial notion. And it was never an issue to us. We wanted it to work on the industry standard. At the time, we were buying a chip from a little company called Seek that did a key part of the Ethernet stuff. So our notion always was to use these standard protocols, TCP/IP and to use Ethernet which was standard, and you know, there was Arcanet, there was XNS, there was IBM's token ring nonsense. There was all this other stuff floating around that didn't make any sense long term. Not because they weren't technically interesting, but because they weren't available. They weren't set up in a way that people could invest in them, and for most of the networking protocols, they weren't set up in a way that would scale. So for either business or technical reasons, those were the right choices. To us this all seemed obvious. But it was a number of years, before you know, IBM at one point had a policy that they could pick any network. Anybody could use any network for anything as long as it wasn't Ethernet, and the reason was that Ethernet was open and they didn't want it -- that it was clearly such anathema to the business people that they basically mandated that you couldn't use the only thing that ended up being widely adopted. So that was clearly a mistake.

I think it was a combination of TCP being open, source code being available, TCP being needed to run well on Ethernet, and also when we did the file system and the way UNIX was done, that people really wanted to interconnect their machines to multiply up their ability to sell into accounts. The file system sharing began. Then the electronic mail became a way of integrating, doing system integration. So we saw a widespread use of TCP with things like NFS that really propelled it and kept going until really the Web kicked it into high gear.

See I'm trying to make the case that it was very gradual and that TCP sort of just spread

and because I've got this whole grass roots thing.

Lots of groups got copies of it. Long. Had it for long -- just dated.

Right, exactly. And that in the mean time, you know because there was this big fight with the OSI versus TCP --

Right. I would have bet that OSI would have overtaken it eventually. I under estimated the grass root strength.

Exactly. And that's exactly the point I want to make in the book. It's kind of the theme of the book.

It doesn't hold well for Microsoft.

No, it doesn't.

No, because it shows that you can't --. OSI had the way to the American governments, European governments, people who said, "I will not procure your equipment and its uses." But it wasn't done grass roots and it wasn't subject to a selection process where a bunch of cantankerous people -- whatever -- whatever you're going to tell that part of the story. That seems to have worked better and it surprises me that it overran. It's not surprising it overran like IBM's proprietary protocols and stuff but that it overran the ISO process. which it clearly did. It was quite amazing. So it's not something you could have organized. It was kind of like a guerilla.

Right, exactly. Like Ben tells these great stories of some guy at some ISO meeting, saying -- kind of dismissing Ben and all these guys -- saying, *"Oh, these university experimenter will go away."*

Right, right. The disdain probably helped too. Cause it means they left us alone.

Right, exactly.

So, it's like, *"Hi, we're the government. We're here to help."* I mean, if they had

come to help, they would have probably ruined it. So there's nothing wrong with ISO protocols, it's just that no one ever cared. Just because the government said that was the right way to do it, it didn't happen.

Well, the point of Postel makes is the fact that TCP has been implemented everywhere and OSI really has really hadn't.

Right, there was one implementation of TCP which was kind of a reference.

Of TCP?

Right. The one that I did. If they had done a reference of implementation of ISO and given it away, otherwise what happened was they had ten vendors implementing it, and the specs weren't clear enough, and they could never get it all sorted out.

Right. You mean OSI?

Uh-huh.

Right.

You end up with this matrix of *"Can I talk to you? I can talk to you, but you can't talk back to me."* Because I mean, it's all too weird. They have this 100 page document. They were doing the same thing with Java. You know it's like there's this language spec and the spec is fine and we are working real hard and getting it correct, but the honest truth is it's this free compiler that you can get off the Net and it kind of defines the truth. You can always check. And without a high quality thing that's available either free or very inexpensively, that's a reference, it's very, very hard to get things to converge to a standard.

So, and that's what TCP was. This free reference.

Yeah. Once it got sent up from Berkeley. Before that it wasn't. Because you got to remember that it was BBN with a guerilla attitude about it. I mean BBN, people are probably still mad at me for the fact that I just took their implementations and said this is garbage. We rewrote it, and then we basically gave it away. That's

embarrassing to them because they were the Internet Gods. Right? But, that was kind of the Internet attitude is like, you know, *"Well, we're just going to make it better."*

Who did you work for the BBN, do you remember?

Well, the guy at BBN had written -- one of the people involved in it was a guy named Rob Gerwitz(?) and there's a guy named Alan Nemeth, who works I think at Dec or Prime or something now, maybe Computer Vision. He was at BBN also. They just weren't happy. But they didn't really see --. See, I'd figure it out from us sending tapes out at Berkeley and getting everyone to send in $50. It was kind of mass marketing at the time. There weren't that many machines out there. You know, there were thousands of axis and we had hundreds of them running software which was done on the University and just mailed out in the wee hours of the night in manilla envelopes. So, you know, we had incredible market share. You know, probably a billion dollars worth of axis running this non-production quality, non-commercially supported. See I mean, all that stuff doesn't really matter if the thing -- if it ships with what we call the *it works option*. Well, yeah, we're not a commercial organization. We don't have support, but we use it. Would you rather use what we -- or would you rather go through this commercial organization which, see that's the thing about TCP/IP, well it wasn't Dec Net or it wasn't whatever but hey.

SNA or --

Yeah, right. SNA, 50 billion pages of specs. What is was, was it worked and you could get the source and you can make it better, and that's the whole --

So you could get the source and tinker with it? Although, but if you did, you were running the risk that it wouldn't talk to other networks.

Well, a lot of the bugs though were boundary conditions.

Were what?

Boundary conditions. It's really an analog system in a lot of ways, in that I send you a packet. You send me a packet back. But I have to estimate how long that's going

to take you to decide when to send another packet if I think maybe you didn't get it. It's hard for me to tell. There's no way for me to tell like how often the packets are being lost or to really exactly measure the characteristics of the channel between us. I don't really know and that can vary over time. So it's what's called historesis. These approximation algorithms. There's this thing called the *silly window syndrome* which is a very famous technical thing, and it's basically, you know, just a breakdown in communication where the thing just confuses itself into doing really stupid things. And these things arise when the ratio's between different things get out of hand. You know, it's like something that was designed to operate in a range of 1 to 3, suddenly it's got a 1 and 1,000, and some mixture of 1 and 1,000 operating against it. Like somebody's going on a land line and somebody's going on a satellite, and it's trying to measure something and it's always wrong. It's like scissors, rock, paper. You know? And if you ever play people that are really good, they get you every time. And that's kind of what happened. So silly windows syndrome was like the worst case behavior where you'd lose like 99% of your performance. You know, Van Jacobson finally came on to Berkeley and figured out a good general algorithm for fixing it so that didn't happen anymore. But that wasn't in the early code.

The early code being?

The stuff I did. Yeah, and as long as you were in a normal environment, meaning a pure LAN environment or pure Wan environment, it wasn't an issue. But when you start mixing it up and having satellites and stuff, the thing started behaving really badly. You know, it's kind of a *"Help, my cursor's gone"* situation. The system just stops working and wasn't easy to -- the problem had never been documented in the literature.

So TCP benefitted a lot it from since it was so widely available. People could actually go out and discover these things. While the people in the committees were arguing about, you know, I don't know, the French wanted something, the German's wanted something, and it's like they're their goofy politic stuff, when there are real practical problems that needed to be addressed before the thing could actually work, and you could only find out about the problems if you would actually deploy it. And so Darpa's money funded those packet radio and other deployments to really get that sorted out quite early.

What you said reminded me of something. Had IP already had been separated out from TCP?

Yeah. Those were separate RFC's when I saw it for the first time.

And do you remember when that happened? When IP was separated out? Did it seem like a good idea to you?

I never saw it when it wasn't separated. Certainly, the OSI Model had those layers separated. I was as much familiar with the OSI Model as the -- I mean TCP/IP, I couldn't tell you which of those came first, because whenever we used to working along on the system, you know, I mean, *"Okay. Now we got to do a networking. Let's go to the library and figure out what people are doing."* Went down to the library, you know.

You mean the real library?

Yeah, the library right next to the building. We went down there, and *"Okay, networking. Let's just look up networking and check out a whole bunch of books, read a whole bunch of papers. Okay, well seems a little confused, but they get this model and it's okay with layers. I don't really understand what the presentation layer is for."*

You went to the library and got a bunch of books?

Oh, yeah. Just check out the books, you know.

What were the books then?

They were just journals. Journal articles from European journals and stuff. And they had their seven layer model or six layer model or whatever the hell it is. And you know, it's, *"Okay. We can do this. I mean. I don't quite understand what -- I sort of understand what the middle layers are or the bottom and the top row -- are a little confusing, but it probably doesn't matter that much. You only have to write. All I have to do is figure how to write the codes. So it doesn't, you know --. We and abstract from this. Just get going, you know. Networking is not a file. We'll figure out sockets." Okay.* And so there were a lot of opportunity to just to sit down and design

something that made sense without worrying. I didn't have to argue with anybody, so we just did it. We decided we had the authority to do it. We had our Vax. We could change the system and then we could mail it to people. So who's to say no. So we just did it.

Right.

Didn't as the Internet people either.

The Internet people being who?

ISI. But you know, I didn't particularly care. I mean, they could write whatever they wanted in their protocol documents. I was only going to ship it if it worked.

[Skipped portion you requested.]

I figured out a way without breaking, because I couldn't change all the implementations. I just did something which they said in one sentence wasn't legal, but I said, *"Oh, it doesn't matter, I don't care."* I just tested it. So I put in keep alives so I could --

What's it called?

Keep alive. Because I wanted to be able to tell if I was logging another machine and the machine went down. I wanted after, you know, just to say it seems to be dead. But their notion was military. Well, we're talking to that tank, and we're never going to give up. The tank gets, you know, you have to just recover at a higher level. Well, that's fine in theory, but not real fine in practice. So, we just changed it. And you know, they had their Telnet program, which was a pain in the butt to use.

They being ISI?

Yeah. They ended up doing a standard Telnet program. I just wanted like a UNIX to UNIX log in command. I wanted a very simple command where I could say, *"Copy A to machine B,"* you know. So I could type a command. I didn't want it to go, *"FTP, blah, blah, blah."* So we made these simple commands. We didn't ask.

We just did it. That improved the creature comforts a lot. I wish we would have thought of mosaic kind of stuff. That wouldn't have been --. There were certainly a lot of ideas like it floating around. We just never did that.

Did the stuff that you guys just did get eventually written into RFC's?

Probably. I never saw them. I didn't worry about it. I know it's built in.

[Skipped portion you requested.]

There's a certain benefit to just being the simplest possible thing. So, there is certainly more elegant -- there are other things that are more elegant, but they're not necessary. That's what the ISO died on. They died on elegance as opposed to practical.

So, let me get this straight. The Telnet that everybody else came out with, an FTP, and then eventually SMTP, were they all part of the stuff that you had?

Eric Allman did a mail program to work this SMTP. He adapted the one that he had already written. Because he already had his program working with UUCP and he adapted it to SMTP. Telnet and FTP did all these things with PDP-10's and funky machines and all these funny nodes. They just didn't make any sense on UNIX which is all E-bid(?) ASCII. We just made a much simpler one. We did use SMTP essentially as it was. And then BBN had their own routing protocols. We made up a very simple one which we sort of borrowed from an XNS spec that we had lying around, which pissed them off enormously because --

You mean Xerox?

Yeah. But this pissed BBN off because they were the ones who were supposed to own the network and all the routing, and now we had to do some routing for our own. But the Berkeley campus needed to have its own routing within the campus, and we weren't going to hire BBN to put their slow switches in, so we just did it using our machines and made up a very simple protocol that we stole out of this manual.

So it was like this end run around BBN.

To a large extent it was, because they were really a bottleneck at that point. Much like I think Microsoft is a bottleneck for innovation in the computer industry right now. And the nice thing is there's this thing, you know, the end to end notion in networks is that if the network allows you and I to communicate and just gets out of the way, that's why they switched away from NCP is the network was trying to add value. Just give me a data channel between the two of us and then we can do whatever we need.

Is this the hop by hop versus end to end reliability that they talk about.

Uh-huh.

If I understand the big difference between NCP and TCP correctly, that the reliability as you shifted from the network which was were it was with NCP to the host. Is that right?

Right. Right. But I think with the reemergence of real time media in the network now, people are starting to think about putting some of the reliability back in because if you want to do audio conferencing especially, you need no hiccups. They've kind of come full circle.

[Skipped portion you requested.]

So with TCP with this reliability issue, if you lose a packet through the network, you just send another one. Is that what happens?

Well, if they lose a packet, you send another one, and it's not selective. One of the bad things about TCP is if I send you 65,000 bytes and 65 packets and the first one gets lost, I have to resend the whole thing.

Oh really?

It only has a notion of what the last thing received is. It doesn't have a notion of what's called selective retransmit. So you can't selectively send back the one thing you lost. So it's very simple minded in that way. XNS was better. That's proved

to be good enough. If you get to really limit conditions like really high speed networks, it gets to be a problem, because that amount can be so large, because you're going so fast. It's like if you're going incredibly fast and you miss an exit, you can't like back up. You're going miles past it. You have to take a very big loop to get back to where you were.

That's happened to me, yeah. And that's TCP?

That's the way it works. Yeah, it's a problem. It's a problem especially for long haul, high speed, long distance communication. In the bandwidth distance product, you take the speed of the link times the physical distance and it's the product of those two numbers that gets very large. Which means that the wire or the pipe that you have in the tube gets to have a lot of capacity. So like I've got an oil pipeline and I've only got a valve to shut it off at the source and I get a leak. All the oil in the pipeline is going to leak out, and that's kind of what TCP is. All that is going to be lost because I don't have the ability to just let the one little thing leak out. I lose it all. I have to send it all again essentially. That's the way that Russians seem to deal with their oil spills in Siberia from what I've read. It's like *"Well, we know it's leaking out in the tundra but we can't turn it off because the pipe would burst, so, we're making an oil lake in the middle of some place." "Oh well. Okay." "Sorry."* Massive function of the weather up there too. But it's a complete disaster.

Is it terribly inefficient and disastrous?

Waste's a lot -- it's bad for the ecology and also wastes a lot of oil too. You've got to keep the oil running just to keep the pipe from freezing. So that's not ideal, but it's very difficult to fix. So the innovation now is really directed at trying to fix a lot of these things without changing things because the cost of change is high.

[Skipped portion you requested.]

That was a 4-processor 360. They had written their own. I guess everybody wrote their own operating system in those days.

Seems that way.

Yeah. I can't think of anybody who didn't.

So you were really into operating systems?

Well, only by choice. Not by choice, but because we at Berkeley, we needed to make it better because we were killing it basically trying to run our large applications. So we were just trying to make our lives bearable by putting in a virtual memory and connecting up more machines. We were having fun.

It's hard to imagine what it must have been like. I mean did you have any sense that --

It was late nights. Like your title.

A lot of late nights.

No. We don't have -- I never have -- I don't really -- no. It wasn't Paris in the 20's. We can romanticize it all we want. It wasn't that much fun. Falling asleep at the keyboard.

Really, would you fall asleep at the keyboard?

Yeah. And then if you lean on the space bar eventually when it gets to the end, it beeps and wakes you up.

Come on.

No really. When it gets to 200 characters then it beeps. With the input of another character it gets mad and so it says beep. So you go, *"Beep!"* Time to go to bed. You'd just fall asleep like this.

What was the reason for staying up so late? Cause you didn't want to leave until you --

You don't get any phone calls. Nobody bothers you. And if it's a time sharing system there's less people logged in, so it goes faster. And also, productivity is really related to the length of time you work on something. Because there's this whole start up and shut down overhead. So if you've got something really hard to do and you

can work on it for ten or twelve hours, you know, you probably get three times as much work done in twelve hours as in eight hours because of that extra four hours, plus you don't have to take time to eat. It is almost uninterrupted time where you've already done all the work to, you know, it's like when I'm writing something, I just try to finish it in one sitting if I can. Because I hate it. It's hell. Or if I write speeches I have to give now, you know, typically a 45 minutes to an hour speech takes me six hours. I just sit with blank slides and I write them over and over again until I get them right and go through the flow and sort it out. You either do it in one sitting or it's -- I had one I wrote over the weekend. It was just hell, because I tried to write it over several days. It just never jelled, because I could never get enough time.

That's sort of the way that the networking stuff was. The socket and all that stuff was basically designed in like one really long day. I was living at my advisor's house, while she was on sabbatical up on Grizzly Peak in Berkeley and just sitting up all night. Start by watching the basketball playoffs which would come in on a tape delay. That was back before NBA was the big thing. The playoffs would come in on tape delay at 11:30 at night. You'd try not to listen to the radio so you wouldn't know the result and then stay up the rest of the night working on writing the spec. It took a couple of days to right the spec. It took us a few years to implement it. So that's the old Edison thing about perspiration and inspiration. One percent inspiration was easy. Getting all that Internet stuff to really work was actually quite a lot of work. Because the trouble with networking code is if it doesn't work, what do you do now?

You're screwed.

Yeah. It's like oh, okay. So these three machines are trying to talk to each other. What I used to do is I'd be on Machine A and I'd say, *"Rlogin B,"* and then I'd say, *"Rlogin A,"* and then I'd say *"Rlogin B"*, and then I'd say, *"Rlogin A,"* so I'd be logged in back and forth and back and forth and back and forth, and I'd start typing and just try doing things. Just to see how many levels deep I could stack it. If you get the machines to do a trace of all the packets that were flying around, it was just unbelievable. But you knew you had it right if you could stack it up like ten or twelve. It's like Captain Crunch and his stacking long distance lines.

Yeah. I love that.

Right. It's the same idea. So we could do the same thing or log in to Boston and back, and Boston and back, and Boston and back, and then see if it still works. So it's pretty easy to test. If it doesn't work, then you scratch your head and try to think of why as opposed to -- you know, because you can sit there and look at reams of trace. You can't tell why it's broken. It's really a hard code to get correct and to perform well, because you have got to be just incredibly careful. I tend to be the kind of person who can think of the things that can go wrong, which is not necessarily a good trait in terms of psychological health, but it's really helpful when you are running that kind of code. Because you almost can't write any code without worrying about all the exceptional conditions. That's kind of what you have to deal with in networking. You have to really make sure you've handled all the corner cases. No dusty corners are allowed.

One of the very early problems --

Programmers are funny. You can know these things are going to happen, but they don't want to deal with it.

Right. Exactly. That's exactly what happened.

Oh it just drives me crazy too, because I can see all sorts of problems and I can't get anybody to pay attention because they are too far away. It's just human nature.

Right. Exactly. It's interesting.

I tried to write the code so you can do what we call correct by construction, which is you try to write -- if you can anticipate enough of the problems, you can structure the code in such a way that many of them can't occur. But then there's the -- but there's certain complexity you can only move around. You can't make it go away. And so it's sort of like picking your poison. You can choose in what strange way -- you often have a choice between two failure modes and you just pick the one that seems least difficult to deal with. Like the difference between hanging forever and telling you that the machine went away or maybe it didn't. And to my mind, hanging forever was worse. Because a lot of times machines do crash and you want

to know. The amount of times it would go away for twenty minutes and then come back were very small. But in the Darpa view, that didn't want to take the chance that it might have gone away, but not really been gone and tell you that. I'd rather take the chance and let you -- you know, so it's a human factors kind of thing.

I'm trying give a sense in the building the IMP chapter about assembly language programming and what that was like. It was a Honeywell 516 thing.

I did assembly language programming in the early 70's.

What is it like?

There's a certain pleasure to it because it's really, you really feel like you're in control of everything. You know, there's always little decisions you have to make. And there's also hack's you could -- used to be able to do to make things go a lot faster. But it's also very easy to make a tiny change and just have the whole thing come apart in your hands. And tracking things down can be really difficult. There's very little predictability. You have to be very careful. A lot of times when you are in a very resource limited environment like they were in, you have to do things which you know if they are not perfect will just leave the thing in a shambles, but you don't have any choice because it wouldn't be fast enough. So it's really a lost art in a way. Game machine people still did it until very recently.

When I look back at the old papers, it says the original IMP had 16K words -- memory?

Well, most machines didn't have bytes. So it's hard to know how many bits there were in the words. And there was probably no way to get something out of memory except as a word.

[Skipped portion you requested.]

In the early days of the Net, we didn't have Crays on the Net. So it was several hundred PDP-10 equivalence sitting on this table, doing all this 3-D razzle dazzle stuff, you know.

That's amazing.

It's really going to change a lot. I'm working on a graphics project right now. We're going to get between a thousand and ten thousand times as much performance as was on those machines in the next ten years.

[Skipped portion you requested.]

They probably don't know what to do with it yet either.

So tell me about what you said earlier about going to that standards meeting when the US got --

I remember that. Well, they wanted to do it from left to right or right to left was the issue. So it's going to be, I guess we had to argue about whether something @ sign something was also a fight too. I don't think that was decided yet. You know like somebody at something. So then you have to decide, let's suppose, so we need a separator. And then there was a big fight about what's on both sides of the @ signs. I think we finally came down to the one side would be the recipient organization, so we decided to split basically your name, which would be the first line of the envelope, from the rest. And then we had to have a fight about what was on each side of the @ sign. And then I remember, one of the big fights on the right side was with the things we'd written from most significant to least significant or whatever. And a lot of people wanted to standardize what was on the left side as well, but they lost, if I remember correctly. And then they came up with this standard on the right hand side where they did like mill and com and all this. And then for the rest it, I think we through in all the ISO two-letter country codes or something. So it wasn't particularly well thought out. Somebody had to make this decision. We spent most of the time fighting about left to right or right to left, which is a very famous thing, you know, because it's like on computers we have this thing called big endian and little endian, basically from the Gulliver's Travel about which end of the egg do you crack things on. There's a similar problem when you take characters and pack them into words. Do you pack them in right to left or left to right in words. The PC and Vax's and PDP-11's were always low to high. And everything else in the world is always high to low. High to low, there's some, I think if you look at it, high to low makes marginally more sense than low to high. Low to high has some funny kind of side effects. But one's slightly better than the other, but neither is correct in any sense. And so I think it's the same thing for this, it's a

social decision. And the important thing was just that there was a meeting. That the decision got made and we went on.

Postel said that he was arguing for the way it is now, the general, and stuff and he thought that it would make --

Oh, yeah. The Tenex guys always love the name completion.

Right. Right.

The problem is that it is hard to take what you have now and then add more stuff on to the end of it. Generally, if you are typing something, if it's written with the most significant stuff first, you can peel off however far you understand and then stop. So if I just add stuff to the end of it written most to least, you only have to understand as far as you understand the other stuff. But if I take least to most and then I want to tack more stuff on, I kind of have to put a separator on like a slash. So you see HTTP:\\blah\, where if it was written the other way around, it could probably be written, HTTP: and just start right in with the most significant. It would save like three or four characters on all the billboards. But we didn't expect to see, as I was driving up the freeway, HTTP:\\WWW.BankAmerica.com, that's all it said. It said Bank of America and it had their URL. That's was all that was on the bulletin board. I was so amazed. It's so weird.

It must be amazing for you guys.

I just kind of decided not to, which was a mistake. I was afraid I would make too many people upset, but it would have been the right thing to do. I was just going to do \\.

[Skipped portion you requested.]

You've got some writing to do. That's pretty scary trying to --

When they came up with the @ sign because it was a line based character?

I hated the @ sign as a line erase character anyway.

As a what?

A line erase character. Because what it did is you typed the @ sign, as you're typing along and type the @, and it just sort of sits there. It didn't like erase the line on the screen or anything. Because it was designed also to work on printing terminals, so it just sort of sat there. It didn't hit return. It didn't echo return or anything. It just couldn't -- I mean, so if you type this, t-h-i-m, instead of m, well, that should have been an s, sharp sign s, oops, I mean to start over, and then @ sign that, you're looking at t-h-i blah sharp sign dno. So you're looking at all that garbage with the @ signs and the # signs. So your just sitting there. You're trying to count the character as you're going backwards to the sharp signs. So we just changed it all so when you hit backspace it would backspace. And when you hit Control U, which meant to kill a line that was to -- Dick decided that I guess -- . Really, if you look at ASCII, there aren't that many characters that aren't used somewhere. @ sign is one of the very few. We haven't really got beyond it like to chinese character set or unicode or anything. There's basically, you've got your basic things across the top of the -- you've got your slash and you less than, your greater than, you're braces and almost all -- . You get into these contexts where almost every single one of those is used. So @ sign was one of the few. They could have used sharp sign. That's really not used very much. But then in England, the sharp sign is the pound character. Or the dollar sign is the pound character and the pound sign is whatever that is that's called the pound. That gets very confusing to what we know, whatever. It's all very confused.

[End of Interview.]